

Files & errors

C Reference

Text files

`fopen` returns a file pointer ; the mode string says what to do — "w" write, "r" read, "a" append. Write with `fprintf`, read with `fscanf`, and always `fclose` when done.

```
#include <stdio.h>

int main(void) {
    FILE *out = fopen("scores.txt", "w");    // open for writing
    fprintf(out, "Alice 80\nBob 95\n");
    fclose(out);

    FILE *in = fopen("scores.txt", "r");    // open for reading
    char name[20];
    int score;
    while (fscanf(in, "%s %d", name, &score) == 2) {
        printf("%s -> %d\n", name, score);
    }
    fclose(in);
    return 0;
}
```

Always close the file



Exam tip: check `fopen != NULL`; mode "r" reads, "w" writes; `fclose` when done

fopen → read/write → *fclose*

Error handling with return codes

C has no exceptions. Instead a function signals failure with a return code — by convention 0 means success and non-zero means an error. The caller checks the code before trusting the result. (`fopen` follows the same idea: it returns `NULL` when it fails.)

```
#include <stdio.h>

// returns 0 on success, -1 if the divisor is 0
int safe_divide(int a, int b, int *result) {
    if (b == 0) return -1;        // error code
    *result = a / b;
    return 0;                    // success
}

int main(void) {
    int r;
    if (safe_divide(10, 2, &r) == 0)
        printf("10 / 2 = %d\n", r);    // 10 / 2 = 5
    if (safe_divide(10, 0, &r) != 0)
        printf("cannot divide by zero\n"); // error reported
    return 0;
}
```

Common mistakes

- Check that `fopen` did not return `NULL` before reading or writing.
- Always `fclose` a file when you are done.
- Return a non-zero code from `main` to signal an error to the caller.